

# 5G-challenge

**ITU-ML5G-PS-015: Network failure detection and root cause analysis in 5GC by NFV-based test environment.**

Kyushu Institute of Technology   Tsukamoto-Lab

TEAM : YOTA-YOTA

M1   Akiyoshi shota   Yamaguchi Yu

B4   Ohshima Saiga   Tashiro Ryouga

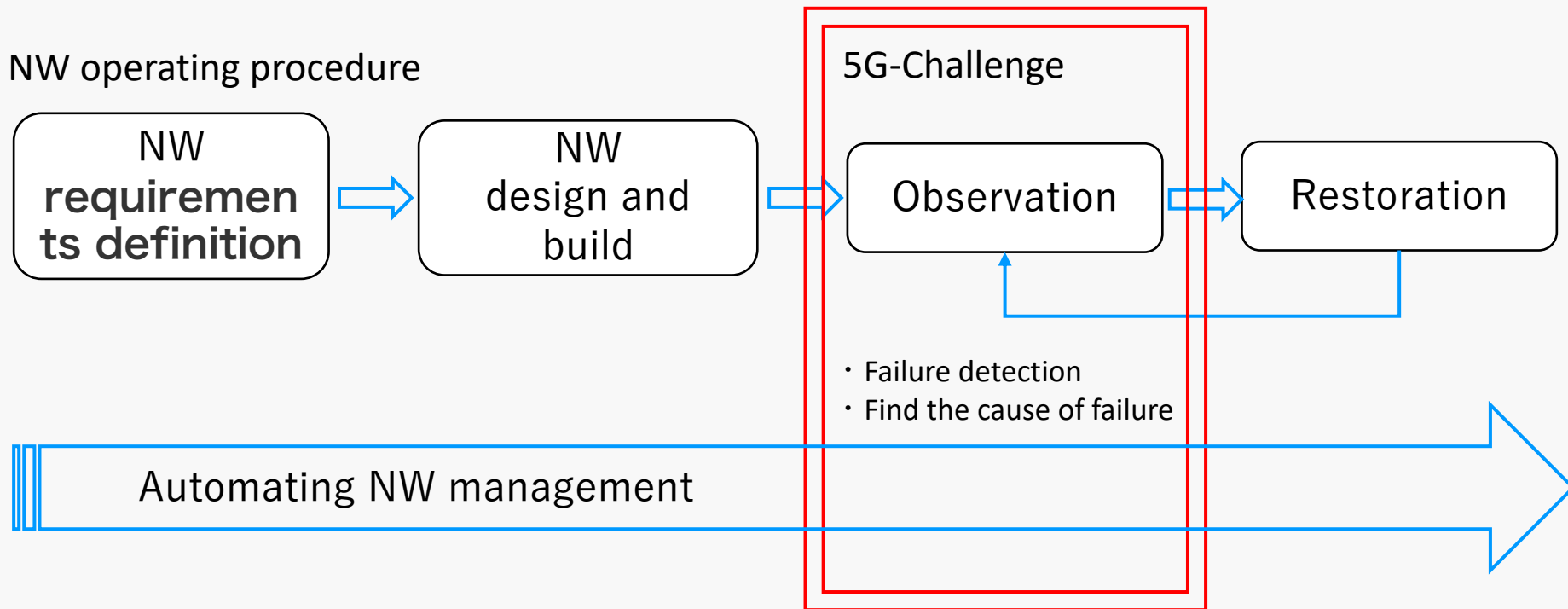
- ❑ Overview of this challenge
- ❑ About dataset
- ❑ Attribute selection and reasons
- ❑ The method of dataset processing
- ❑ Algorithm used
- ❑ Configuration of the learner
- ❑ Results
  - ❑ Accuracy
  - ❑ confusion matrix
  - ❑ Analysis by failure
    - ❑ Precision • Recall • F-Measure
- ❑ Discussion

# Overview of this challenge

## Theme

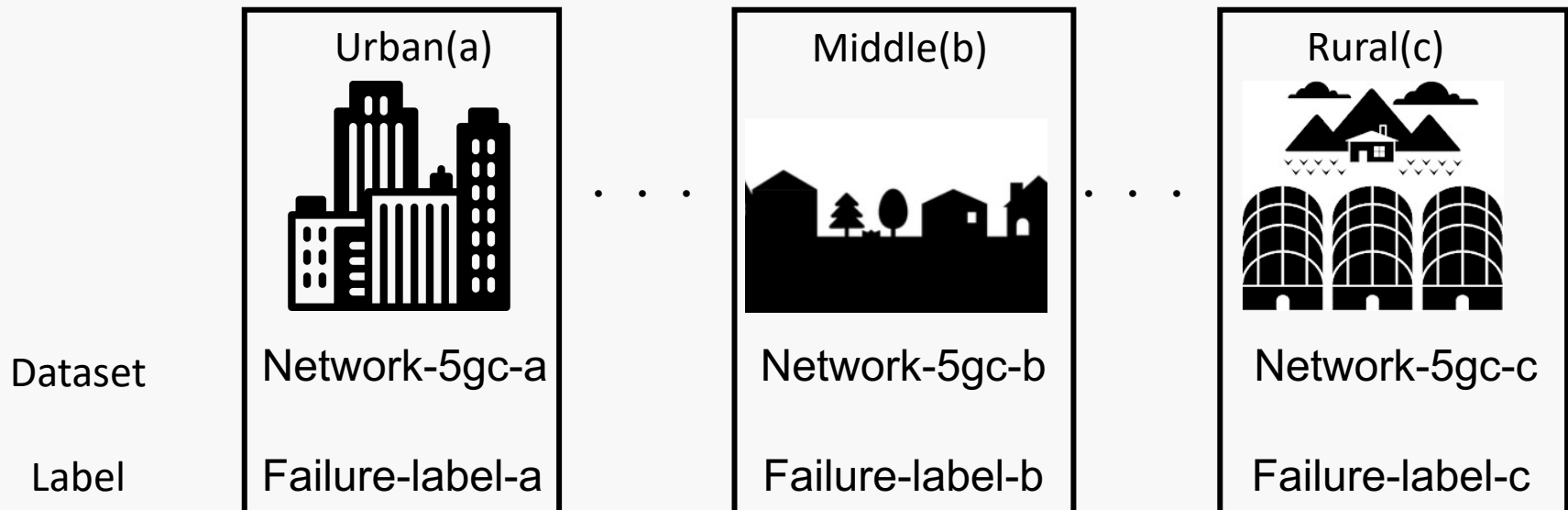
- Network fault detection and root cause analysis within the 5G core network in NFV-based test environment.
- Promote automation of networks using AI and ML.
  - Aiming for accelerating the spread of 5G networks, stable and high quality operations.

NW operating procedure



## ■ Dataset(format : json)

- Challenge 1 : Failure detection at urban(a) and rural(c) area.
  - Labels showing data and faults acquired in the 5G core network in urban (a) and rural (c) areas are given, respectively.
- Challenge 2 : Failure detection at middle(b) area.
  - Labels showing data acquired and failures in the 5G core network in the middle (b) area are also given, **but its amount is quite smaller compared with other areas (a) and (b).**



- Cases are classified into the total of 6(Properly:1 type + Failure:5 types) types.
  - Case 0. **normal**
  - Case 1. **bridge-delif(addif)**
    - **in-octets** : "0" Streak is seen in duration of failure (represent by  $(N+10)-N$ )
  - Case 2. **interface-down(up)**
    - **oper-status** : "up" change "down" under failure(represent by 0 as "up" and 1 as "down")
  - Case 3. **interface-loss-start(stop)**
    - **out-octets** : "0" Streak is seen in duration of failure (represent by  $(N+10)-N$ )
- Each Cases is determined by attribute writen by blue.

- Cases are classified into the total of 6(Properly:1 type + Failure:5 types) types.
- Case 4. **memory-stress-start(stop)**
  - **memory-status** : "Healthy" change "Critical" under failure(represent by 0 as "Healthy" and 1 as "Critical")
  - **used-percent** : Percentage value change 80% level to 90% level under failure(represent by converting percentage value into ratio)
- Case 5. **vcpu-overload-start(stop)**
  - **user** : The value increase under failure(represent by converting percentage value into ratio)
  - **system** : The value increase under failure(represent by converting percentage value into ratio)
  - **idle** : The value decrease under failure(represent by converting percentage value into ratio)

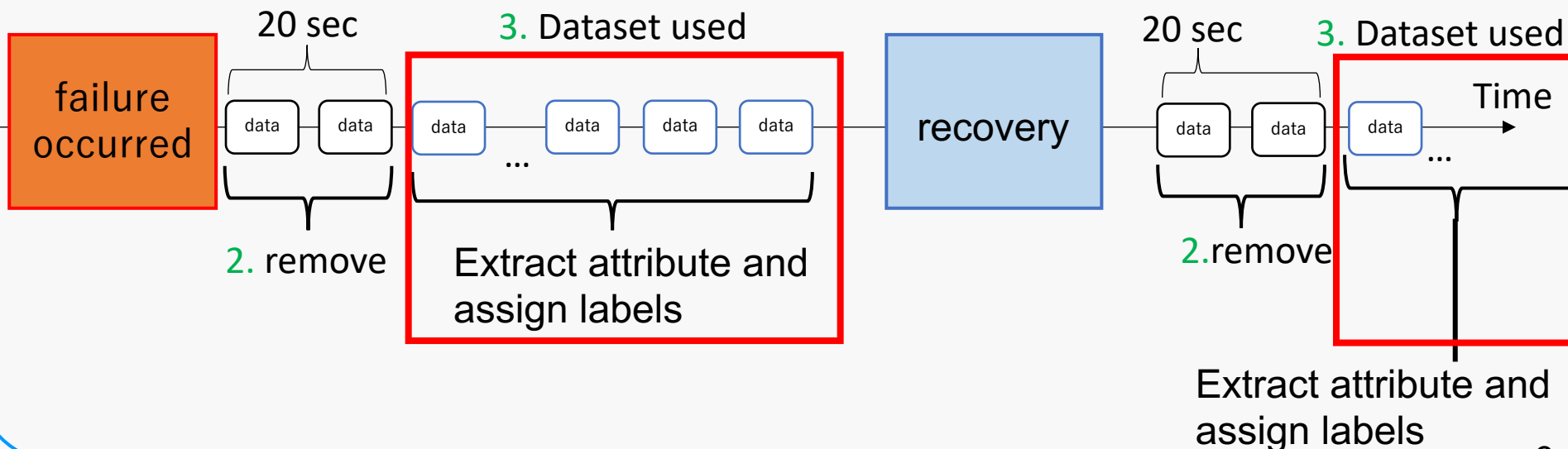
➤ Each Cases is determined by attribute written by blue.

# The method of dataset processing

## ■ Data processing procedure

1. Compares the failure occurrence time of the correct label with the network data acquisition time, and arrange them in chronological order.
2. Remove unstable data within 20 seconds of the time of failure and failure recovery.
3. Use the remaining data as a data set.
  - For these data, extract the chosen attribute and assign labels.

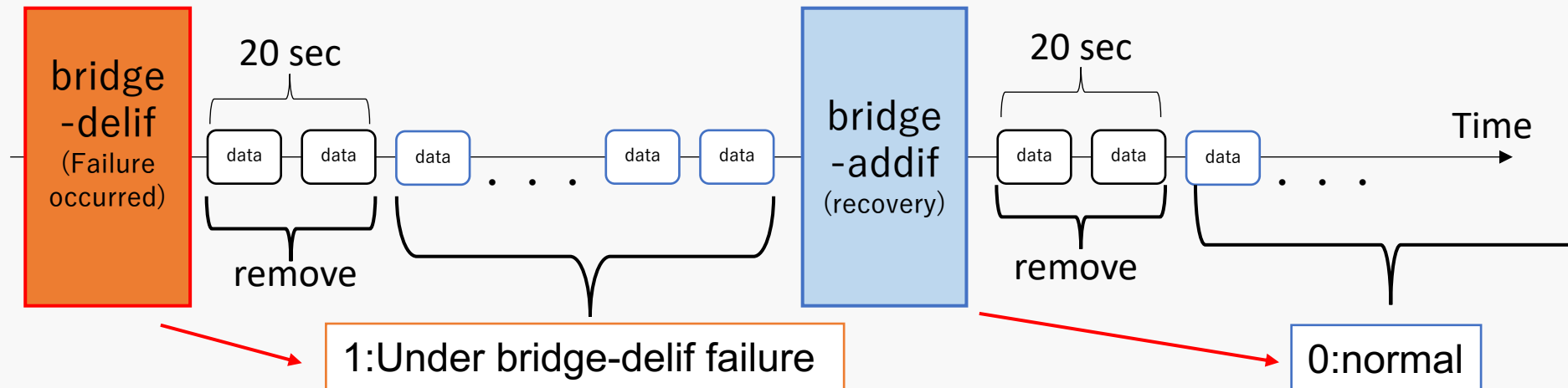
### 1. Dataset line arranged



# How to assign labels

## Assign labels as bellow

- 0:normal (Properly condition)
- 1:Under bridge-delif(addif) failure
- 2:Under interface-down(up) failure
- 3:Under interface-loss-start(stop) failure
- 4:Under memory-stress-start(stop) failure
- 5:Under vcpu-overload-start(stop) failure





## ❑ Problems with the original data

- ❑ Large capacity makes it difficult to use all data as input data  
→ Improve learning speed by reducing data

## ❑ Data processing

- ❑ Output the required features from the distributed json file to a single line of txt.



	original data	Post-processing data
Urban area:a	32[GB]	18[MB]
Rural area:c	32[GB]	18[MB]
middle:b	9[GB]	5[MB]
Total	73[GB]	41[MB]

# Details of post-processing data

- Change in number of data due to deletion of data in transition.

	Area	a	c	b
	Data type	Urban	Rural	middle
original data	Train	98533	98533	12360
	Test	24647	24647	24647
Post-processing data	Train	93505	93515	11740
	Test	23360	23366	23363

Low data volume compared to urban and rural areas

# Details of post-processing data

0:normal  
1: bridge-delif(addif)  
2:interface-down(up)  
3:interface-loss-start(stop)  
4:memory-stress-start(stop)  
5:vcpu-overload-start(stop)

## Breakdown of training data

Area Label	a Urban	c Rural	b middle	
0	87649(94%)	87690(94%)	11000(94%)	normal
1	1226(1.2%)	1230(1.2%)	152(1.3%)	Failure
2	1142(1.2%)	1119(1.2%)	146(1.2%)	
3	1139(1.2%)	1141(1.3%)	145(1.2%)	
4	1172(1.2%)	1157(1.2%)	149(1.3%)	
5	1177(1.2%)	1178(1.3%)	148(1.3%)	
Total	93505	93515	11740	
Failre / Total	6.7%	6.6%	6.7%	

# Details of post-processing data

## ■ Breakdown of test data

0:normal  
 1: bridge-delif(addif)  
 2:interface-down(up)  
 3:interface-loss-start(stop)  
 4:memory-stress-start(stop)  
 5:vcpu-overload-start(stop)

Area Label	a Urban	c Rural	b middle	
0	21900(94%)	21914(94%)	21891(94%)	normal
1	308(1.3%)	306(1.3%)	312(1.3%)	Failure
2	287(1.2%)	278(1.2%)	286(1.2%)	
3	280(1.3%)	284(1.2%)	287(1.2%)	
4	291(1.2%)	292(1.2%)	295(1.3%)	
5	294(1.3%)	292(1.2%)	292(1.2%)	
Total	23360	23366	23363	
Failre / Total	6.7%	6.6%	6.7%	

# Consideration of algorithms to be used

■ For each task, the two algorithms were compared.

■ Selection criteria

- Good at classification and regression
- Capable of learning with high accuracy
- Many salient features in the data set

■ Algorithms

■ XGBoost

- A method that combines ensemble learning and decision trees called gradient boosting.

■ LightGBM

- A modified version of XGBoost
- Faster and more accurate learning than XGBoost

*dmlc*  
**XGBoost**

 **LightGBM**



Adopt highly accurate algorithms.

## Task1 : Failure detection at urban(a) and rural(c) area

- Pattern 1 : Classification of test data a.

Algorithm	XGBoost	LightGBM
Accuracy	0.96609589	0.96091609

- Pattern 2 : Classification of test data c.


Algorithm	XGBoost	LightGBM
Accuracy	0.984165026	0.98219635

## Task2 : Failure detection at middle(b) area

Algorithm	XGBoost	LightGBM
Accuracy	0.972178231	0.96819757

## Algorithm selection

- We selected algorithms based on classification accuracy.
- XGBoost had better accuracy in all tasks.

 XGBoost is adopted.

# How to use the data set

## Task1 : Failure detection at urban(a) and rural(c) area

### Candidate training models

- Share model : Training with all data from both a and c
- Individual model : Create a separate trainer for each of a and c

→ Determine the use of a common model based on the percentage of correct answers.

	Share model		Individual model	
Training data	a , c	a , c	a	c
Test data	a	c	a	c
Accuracy	0.96609589	0.984165026	0.96160102	0.98335187
	Pattern1	Pattern2		

## Task2 : Failure detection at middle(b) area

- To improve accuracy by increasing the amount of training data
- Consider using the data for learning task 1.

Training data	b	a , b	b , c	a , b , c
Accuracy	0.96943885	0.972178231	0.96901082	0.96866840

## Task1

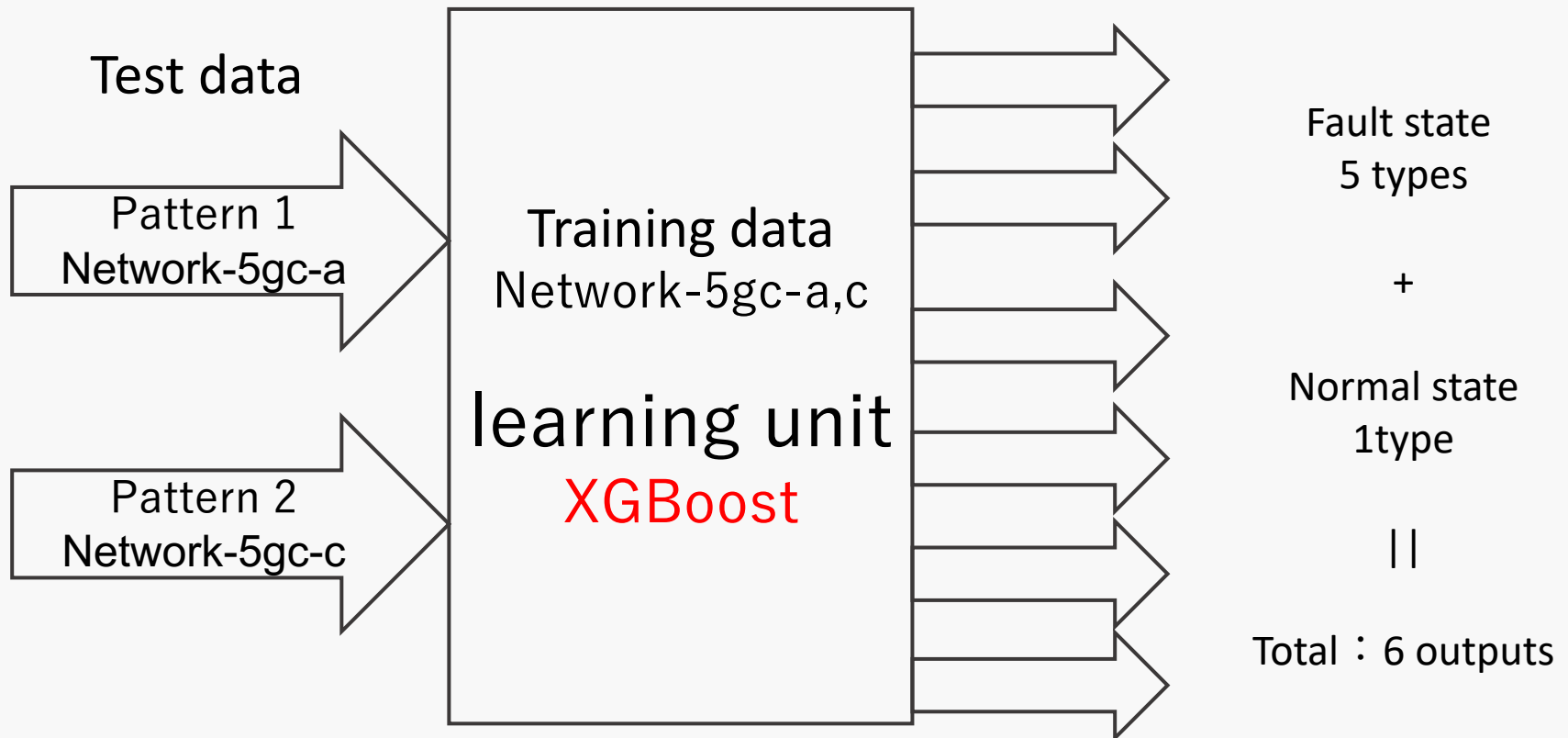
Training data : Network-5gc-a,c

Pattern 1

Test data : Network-5gc-a

Pattern 2

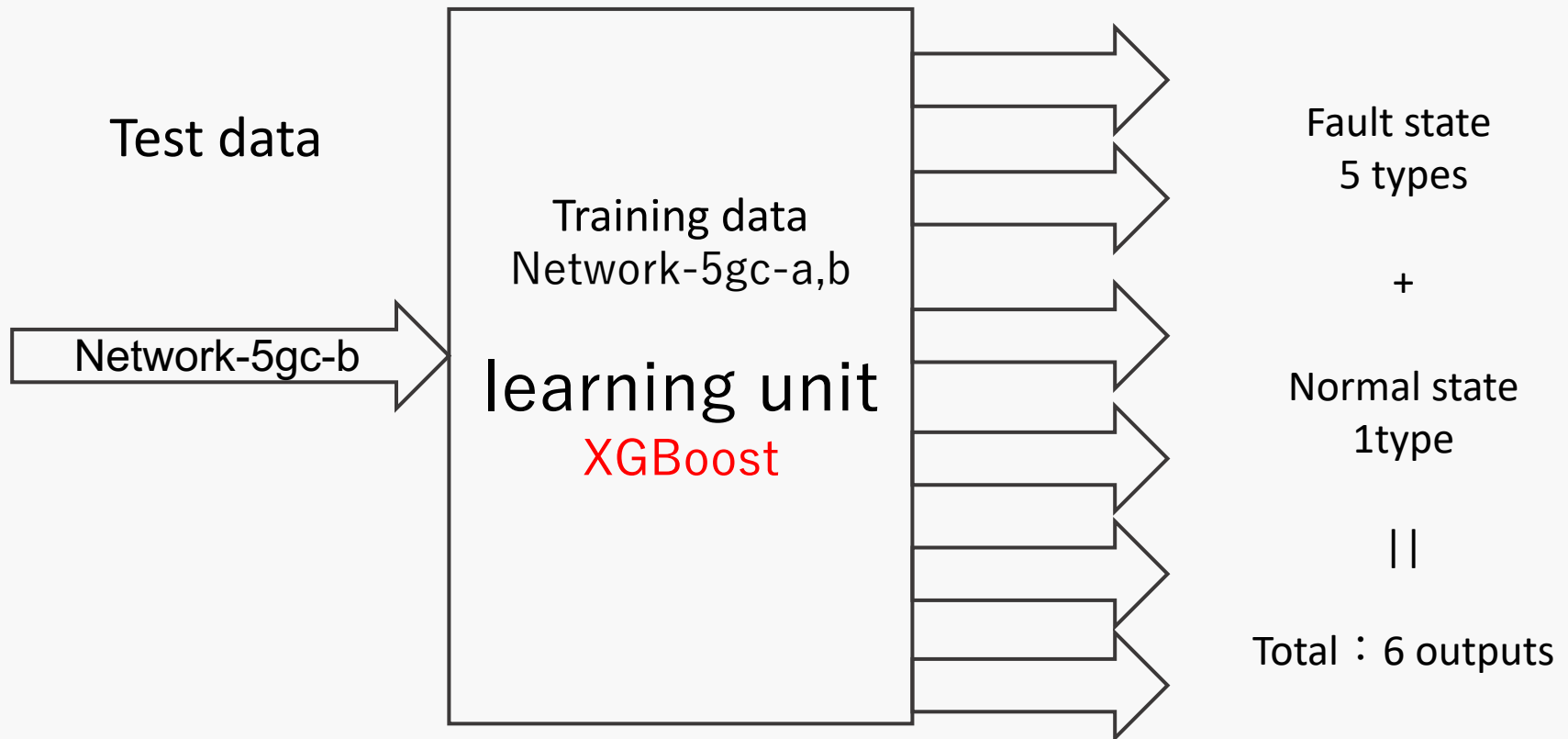
Test data : Network-5gc-c





## Task2

- Training data : Network-5gc-a,b (Task 1: Borrowing data from a)
- Test data : Network-5gc-b



# Result: Task 1, Pattern 1

Training Data  
Urban(a) Rural(c)

Test Data  
Urban(a)

## Accuracy

96.61%

0:normal  
1: bridge-delif(addif)  
2:interface-down(up)  
3:interface-loss-start(stop)  
4:memory-stress-start(stop)  
5:vcpu-overload-start(stop)

## Confusion matrix

predict \ True	0	1	2	3	4	5
0	21354	15	519	11	1	0
1	82	222	0	4	0	0
2	0	0	287	0	0	0
3	137	23	0	120	0	0
4	0	0	0	0	291	0
5	0	0	0	0	0	294

# Result: Task 1, Pattern 1

Training Data  
Urban(a) Rural(c)

Test Data  
Urban(a)

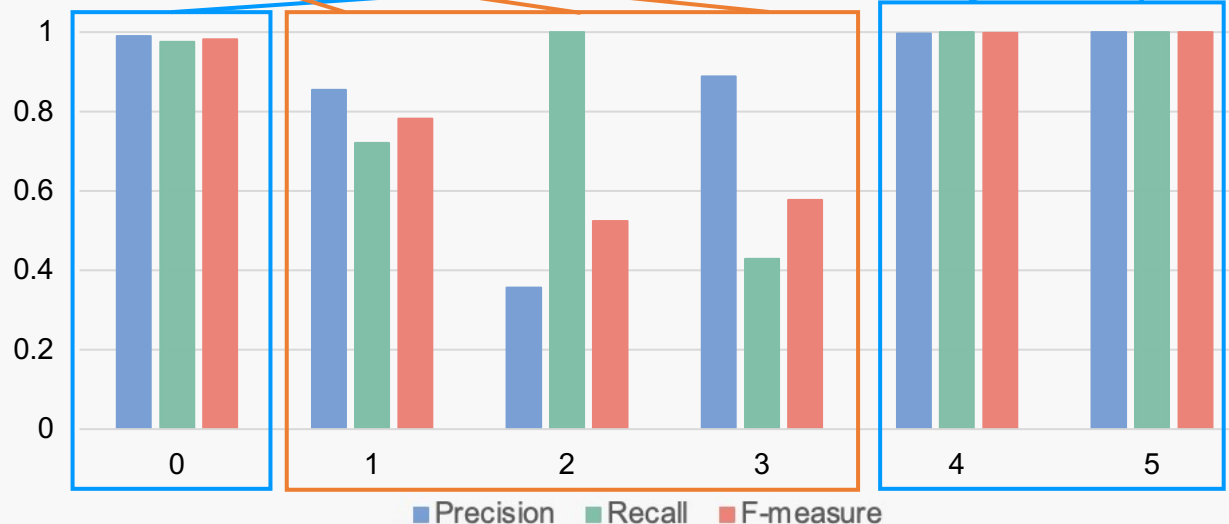
## Failure analysis

0:normal  
1: bridge-delif(addif)  
2:interface-down(up)  
3:interface-loss-start(stop)  
4:memory-stress-start(stop)  
5:vcpu-overload-start(stop)

	0	1	2	3	4	5	Ave.
Accuracy	0.96609589						
Precision	0.990	0.854	0.356	0.889	0.997	1.000	0.848
Recall	0.975	0.721	1.000	0.429	1.000	1.000	0.854
F-measure	0.982	0.782	0.525	0.578	0.998	1.000	0.811

- Failure 1, 2 and 3 have low precision, recall, and F-measure.
- The current attribute cannot classify with high accuracy.

- Normal and failure 4 and 5 have high precision, recall, and F-measure.
- The current attribute can classify with high accuracy.



# Results: Task 1, Pattern 2

Training Data  
Urban(a) Rural(c)

Test Data  
Rural(c)

## Accuracy

98.42%

0:normal  
1: bridge-delif(addif)  
2:interface-down(up)  
3:interface-loss-start(stop)  
4:memory-stress-start(stop)  
5:vcpu-overload-start(stop)

## Confusion matrix

predict True \	0	1	2	3	4	5
0	21884	9	0	21	0	0
1	121	180	0	5	0	0
2	0	0	278	0	0	0
3	209	5	0	70	0	0
4	0	0	0	0	292	0
5	0	0	0	0	0	292

# Results: Task 1, Pattern 2

Training Data  
Urban(a) Rural(c)

Test Data  
Rural(c)

## Failure analysis

	0	1	2	3	4	5	平均
Accuracy	0.984165026						
Precision	0.985	0.928	1.000	0.729	1.000	1.000	0.940
Recall	0.999	0.588	1.000	0.246	1.000	1.000	0.806
F-measure	0.992	0.720	1.000	0.368	1.000	1.000	0.847

0:normal

1: bridge-delif(addif)

2:interface-down(up)

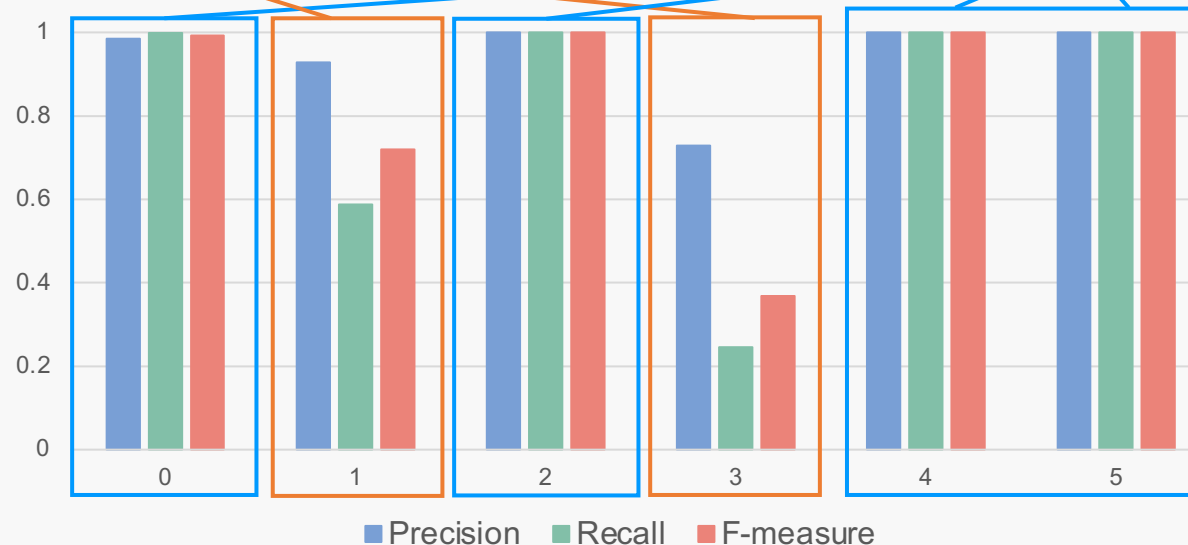
3:interface-loss-start(stop)

4:memory-stress-start(stop)

5:vcpu-overload-start(stop)

- Failure 1 and 3 have low precision, recall, and F-measure.
- The current attribute cannot classify with high accuracy.

- Normal and failure 2, 4 and 5 have high precision, recall, and F-measure.
- The current attribute can classify with high accuracy.



# Results: Task 2

Training Data  
Urban(a) middle(b)

Test Data  
middle(b)

## Accuracy

97.22%

0:normal  
1: bridge-delif(addif)  
2:interface-down(up)  
3:interface-loss-start(stop)  
4:memory-stress-start(stop)  
5:vcpu-overload-start(stop)

## Confusion matrix

predict \ True	0	1	2	3	4	5
0	21849	6	0	32	4	0
1	270	38	0	4	0	0
2	95	0	191	0	0	0
3	235	4	0	48	0	0
4	0	0	0	0	295	0
5	0	0	0	0	0	292

# Results: Task 2

Training Data  
Urban(a) middle(b)

Test Data  
middle(b)

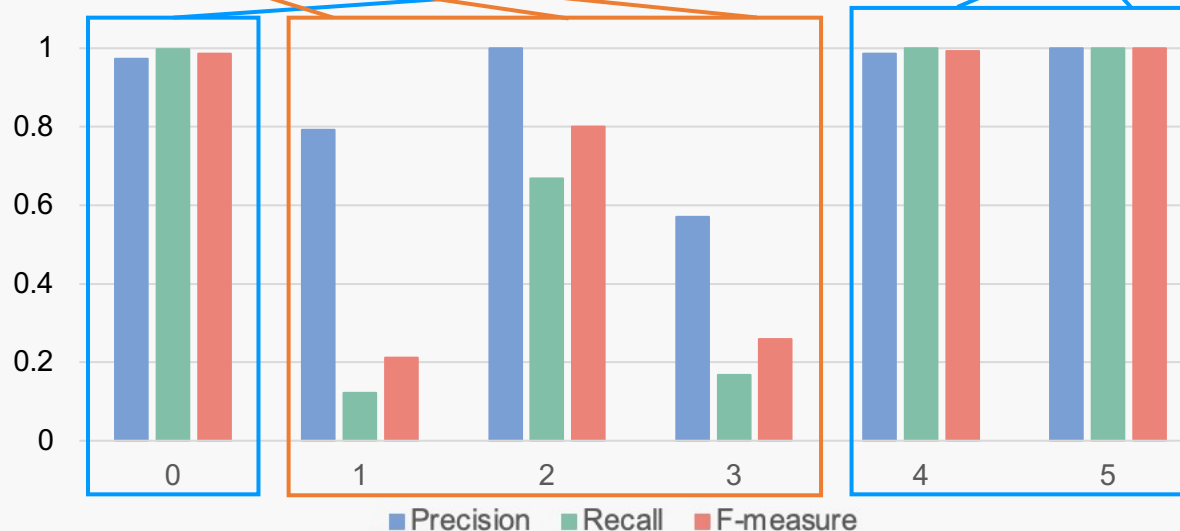
## Failure analysis

	0	1	2	3	4	5	平均
Accuracy	0.972178231						
Precision	0.973	0.792	1.000	0.571	0.987	1.000	0.887
Recall	0.998	0.122	0.668	0.167	1.000	1.000	0.659
F-measure	0.986	0.211	0.801	0.259	0.993	1.000	0.708

0:normal  
1: bridge-delif(addif)  
2:interface-down(up)  
3:interface-loss-start(stop)  
4:memory-stress-start(stop)  
5:vcpu-overload-start(stop)

- Failure 1, 2 and 3 have low precision, recall, and F-measure.
- The current attribute cannot classify with high accuracy.

- Normal and failure 4 and 5 have high precision, recall, and F-measure.
- The current attribute can classify with high accuracy.



## ■ Comparison of Algorithms

- XGBoost gave higher accuracy results than LightGBM.
  - LightGBM : Leaf-wise (A method of prioritizing leaves for branching)
  - XGBoost : Level-wise (A method for branching without prioritizing leaves.)  
→ There is a difference in the direction of learning, and the Level-wise direction was appropriate for this assignment.

## ■ How to use the data set

- Task1 : The common model had higher prediction accuracy.
  - The common model was superior because the characteristics of urban and rural areas were the same and the same features could be used.
- Task2 : The model with the addition of data from the urban area (a) is more accurate.
  - The data from rural areas (c) may contain noise that prevents the model from learning well for the middle part of the model.
  - Data from urban area (a) can be used to increase the amount of training data and improve accuracy.



## □ About the results

- Failure with feature that show obvious changes with failure(failure 4 and 5) can be classified with high accuracy.
- It is difficult to classify at high accuracy when the changes in feature values seen in the failure are also seen in the normal condition (failure 1 and 3).
- To achieve realistic failure detection.
  - We need a predictor with high recall to avoid missing failures.
    - Examples of predictions to avoid : Prediction of normal, but actual failure

## □ Points of Improvement

- Improvement can be expected by searching for features for failure 1 and 3 in the dataset or by discovering new relationships through data processing and using them in training.
- We have not entered the features for normal. Therefore, we believe that the classification accuracy of normal can be improved by adding features for the completely normal state.

# The following are supplementary slides

- The following are supplementary slides

## ❑ Comparison of Algorithms

- ❑ XGBoost gave higher accuracy results than LightGBM.
  - ❑ LightGBM : Leaf-wise (A method of prioritizing leaves for branching)
  - ❑ XGBoost : Level-wise (A method for branching without prioritizing leaves.)  
→ There is a difference in the direction of learning, and the Level-wise direction was appropriate for this Task.

## ❑ How to use the data set

- ❑ Task1 : The share model had higher prediction accuracy.
  - ❑ The features at failure in urban and rural areas don't change. So, the same features can be used.
- ❑ Task2 : The model with data from the urban area (a) is more accurate.
  - ❑ For the middle part that is to be estimated, the data in the rural part (c) may contain noise that prevents the learning from proceeding well.
  - ❑ The accuracy can be improved by increasing the number of data for training with data from urban areas (a).

## □ About the results

- Failure 4 and 5, which use features that show clear changes during failure, can be classified with high accuracy.
- It is difficult to classify failure 1 and 3 with high accuracy because the changes in features seen in failure are also seen in normal conditions.
- To achieve failure detection in a real environment
  - We need a predictor with high recall to avoid missing failures.
    - Examples to avoid : Prediction of normal, but actual failure

## □ Points of Improvement

- For failure 1 and failure 3
  - Search for features in the dataset.
  - Discover new relationships through data processing and use them for learning.
- We have not included features for normal.
  - We believe that the classification accuracy of normal can be improved by adding features in the case of normal conditions.

Thank you for listening!

# The following are supplementary slides

- The following are supplementary slides

## ▣ Feature extraction

- ▣ dataset.py : Display the type and number of failures
- ▣ fault\_location.py : Display the type of failure and the location of the failure
- ▣ relate\_data\_time.py : Display labels (faults) and data in chronological order
- ▣ label.py : Display of absolute path and fault type
- ▣ relate\_data\_time\_ctm\_yamaguchi : Display the correct label file name, fault type, and fault interface.

## ▣ data preprocessing

- ▣ key.py : Extracting feature data from a json file
- ▣ 10sec\_inoctet : Display of values for 10 seconds and time series of data and correct answer labels
- ▣ exlude\_non\_parama\_data.py : Exclude files with missing data
- ▣ make\_dataset : Dataset creation (data extraction, preprocessing, correct answer labeling)

## ▣ Learning phase

- ▣ custom\_xgboost : Learning with xgboost
- ▣ test\_xg\_light.py : Learning with light\_gbm

Model name / type	CERVO Deep for Linux GP-I910900XA3N480TSDDe
OS	Ubuntu18.04LTS
Framework	CUDA10/Tensor Flow/Keras/Chainer for Linux
Chip set	Intel X299 Chip set
Processor	Core i9 10900X (10core/20thread/3.7GHz/tb4.5GHz/19.25MB/165W) LGA2066, 14nm Max256GB DDR4-2933
Memory	256GB (32GB × 8) DDR4-3200(PC4-25600)
Storage (standard)	480GB SSD 6Gb/s SATA R:550MB/s W:520MB/s MTBF: 2 million hours High endurance SSD Samsung PM883 series
Storage(expansion)	2TB(7200rpm, 128MB, 6Gb/s SATA) MTBF=2 million hours High Endurance HDD
Graphics (drawing, calculation)	GeForceRTX3090 24GB GDDR6x (DisplayPort × 2,HDMI × 2) TDP=350W



Library Name	Description.
xgboost	An open source library that implements the gradient boosting decision tree algorithm.
scikit-learn	Libraries for machine learning
optuna	Library for hyperparameter optimization
pandas	Libraries for data analysis
numpy	Libraries for numerical operations
matplotlib	Library for plotting data on a graph

# Learning Environment: Version Information

Library	Version
xgboost	1.5.0
scikit-learn	1.0.1
optuna	2.10.0
pandas	1.3.4
numpy	1.17.3
matplotlib	3.4.3

# Learning environment: Hyperparameters

Name	Parameters	Description.
objective	multi:softmax	Used in multi-level classification problems. Softmax outputs the class whose prediction has the highest probability.
num_class	6	Number of classes
eval_metric	logloss	Learning metrics for test data logloss is a negative logarithmic scale
max_depth	6	Maximum depth of tree
min_child_weight	1	Lower limit of the weight of the leaves of the decision tree
subsample	1	Percentage of samples randomly sampled in each decision tree
colsample_bytree	1	Percentage of columns randomly extracted in each decision tree.
learning_rate	0.3	Learning rate parameter to prevent overlearning

## □ Recall

- How many of the correct answers were predicted to be correct examples.
- Useful when you want to reduce the number of errors.

## □ Precision

- How much of what you predicted to be correct examples was correct.
- Useful when you want to reduce the number of false positives.

## □ Example

- Diagnosis of positive or negative for 1000 people, with 10 positive and 990 negative.
  - Case 1: Five positive diagnoses, all five of which were positive.
    - Reproduction rate  $5/10 = 0.5$
    - Goodness of fit  $5/5 = 1$
  - Case 2: There were 50 positive diagnoses, and 10 of them were positive.
    - Reproduction rate  $10/10 = 1$
    - Goodness of fit  $10/50 = 0.2$
  - Case 1: 0 false positives , 5 failures  $\leftrightarrow$  Case 2: 40 false positives , 0 failures

# Result: Task 1, Pattern 1

## Accuracy

96.61%

## Confusion matrix

Predict \ true	0	1	2	3	4	5
0	21354	15	519	11	1	0
1	82	222	0	4	0	0
2	0	0	287	0	0	0
3	137	23	0	120	0	0
4	0	0	0	0	291	0
5	0	0	0	0	0	294

0:normal  
1: bridge-delif(addif)  
2:interface-down(up)  
3:interface-loss-start(stop)  
4:memory-stress-start(stop)  
5:vcpu-overload-start(stop)

## Failure analysis

	0	1	2	3	4	5	平均
正解率	0.96609589						
適合率	0.990	0.854	0.356	0.889	0.997	1.000	0.848
再現率	0.975	0.721	1.000	0.429	1.000	1.000	0.854
F値	0.982	0.782	0.525	0.578	0.998	1.000	0.811

# Result: Task 1, Pattern 2

## Accuracy

98.42%

## Confusion matrix

予測 正解	0	1	2	3	4	5
0	21884	9	0	21	0	0
1	121	180	0	5	0	0
2	0	0	278	0	0	0
3	209	5	0	70	0	0
4	0	0	0	0	292	0
5	0	0	0	0	0	292

0:normal  
1: bridge-delif(addif)  
2:interface-down(up)  
3:interface-loss-start(stop)  
4:memory-stress-start(stop)  
5:vcpu-overload-start(stop)

## Failure analysis

	0	1	2	3	4	5	平均
正解率	0.984165026						
適合率	0.985	0.928	1.000	0.729	1.000	1.000	0.940
再現率	0.999	0.588	1.000	0.246	1.000	1.000	0.806
F値	0.992	0.720	1.000	0.368	1.000	1.000	0.847

# Result: Task 2

## Accuracy

97.22%

## Confusion matrix

予測 正解	0	1	2	3	4	5
0	21849	6	0	32	4	0
1	270	38	0	4	0	0
2	95	0	191	0	0	0
3	235	4	0	48	0	0
4	0	0	0	0	295	0
5	0	0	0	0	0	292

0:normal  
1: bridge-delif(addif)  
2:interface-down(up)  
3:interface-loss-start(stop)  
4:memory-stress-start(stop)  
5:vcpu-overload-start(stop)

## Failure analysis

	0	1	2	3	4	5	平均
正解率	0.972178231						
適合率	0.973	0.792	1.000	0.571	0.987	1.000	0.887
再現率	0.998	0.122	0.668	0.167	1.000	1.000	0.659
F値	0.986	0.211	0.801	0.259	0.993	1.000	0.708